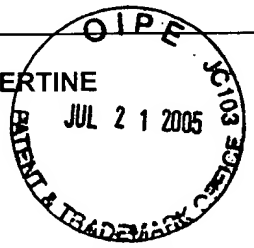
TRANSMITTAL FORM	Attorney Docket No. AUS920000328US1 1752P



In re the application **AUBERTINE** Confirmation No: **1975**

Serial No: **09/903,947** Group Art Unit: **2124**

Filed: **July 12, 2001** Examiner: **Vu, Tuan A.**

For: **Method and System for Minimizing the Cycle Time When Compiling A Program In A Processing System**

ENCLOSURES *(check all that apply)*					
☐	Amendment/Reply	☐	Assignment and Recordation Cover Sheet	☐	After Allowance Communication to Group
	☐ After Final	☐	Part B-Issue Fee Transmittal	☐	Notice of Appeal
☐	Information disclosure statement	☐	Letter to Draftsman	■	Appeal Brief
	☐ Form 1449	☐	Drawings	☐	Status Letter
	☐ (X) Copies of References	☐	Petition	■	Postcard
☐	Extension of Time Request *	☐	Fee Address Indication Form	☐	Other Enclosure(s) (*please identify below):*
☐	Express Abandonment	☐	Terminal Disclaimer		
☐	Certified Copy of Priority Doc	☐	Power of Attorney and Revocation of Prior Powers		
☐	Response to Incomplete Appln	☐	Change of Correspondence Address		
☐	Response to Missing Parts	*Extension of Term: Pursuant to 37 CFR 1.136, Applicant petitions the Commissioner to extend the time for response for xxxxxx month(s), from to .			
	☐ Executed Declaration by Inventor(s)				

CLAIMS					
FOR	Claims Remaining After Amendment	Highest # of Claims Previously Paid For	Extra Claims	RATE	FEE
Total Claims	0	0	0	$ 50.00	$ 0.00
Independent Claims	0	0	0	$200.00	$ 0.00
				Total Fees	$ 0.00

METHOD OF PAYMENT

☐	Check no. _____ in the amount of $ _____ is enclosed for payment of fees.
■	Charge $500.00 to Deposit Account No. 09-0447 (IBM Corporation) for payment of fees.
■	Charge any additional fees or credit any overpayment to Deposit Account No. 09-0447 (IBM Corporation)

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Attorney Name	Joseph A. Sawyer, Reg. No. 30,801.
Signature	
Date	July 19, 2005

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on July 19, 2005

Type or printed name	Irena Nikolova
Signature	

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

APPEAL NO:

In Re Application of:

Matthew E. AUBERTINE

Serial No: 09/903,947

Filed: July 12, 2001

For: METHOD AND SYSTEM FOR MINIMIZING THE CYCLE TIME WHEN COMPILING A PROGRAM IN A PROCESSING SYSTEM

APPELLANT'S BRIEF

Joseph A. Sawyer, Jr.
Attorney for Appellants

Sawyer Law Group LLP
P.O. Box 51418
Palo Alto, CA 94303

TOPICAL INDEX

CERTIFICATE OF MAIL

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on **July 19, 2005**.

Irena Nikolova

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In Re Application of:	Date: July 19, 2005
Matthew E. AUBERTINE	Confirmation No. 1975
Serial No: 09/903,947	Group Art Unit: 2124
Filed: July 12, 2001	Examiner: Vu, Tuan A.

For: METHOD AND SYSTEM FOR MINIMIZING THE CYCLE TIME WHEN COMPILING A PROGRAM IN A PROCESSING SYSTEM

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF ON APPEAL

Sir:

Appellant herein files an Appeal Brief drafted in accordance with the provisions of 37 C.F.R. §1.192(c) as follows:

I. REAL PARTY IN INTEREST

Appellants respectfully submit that the above-captioned application is assigned, in its entirety to International Business Machines Corporation.

II. RELATED APPEALS AND INTERFERENCES

Appellants state that, upon information and belief, they are not aware of any co-pending appeal or interference which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Application Serial No. 09/903,947 (the instant application) as originally filed included claims 1-15. Claims 1-15 are pending. Claims 1-15 are on appeal and all applied prospective rejections concerning Claims 1-15 are being appealed herein.

IV. STATUS OF AMENDMENT

All amendments made to the instant application have been entered.

V. SUMMARY OF THE INVENTION

The present invention provides a method and system for minimizing the cycle time when compiling a program in a computer system, where a program includes a plurality of directories and each of the directories includes a code file. In accordance with the present invention, the method includes: (a) providing a master array of directories of the program, wherein the master array lists the dependencies of the directories; (b) providing a code change to the program to provide an updated program; (c) providing associated dependency changes to the master array to provide an updated master array; and (d) compiling the updated program utilizing the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories. As a result, compile cycle time for large programs is significantly reduced. A second advantage is that since the dependencies are updated

substantially simultaneously with code changes, there are minimal dependency violations and therefore few deadlocks.

VI. ISSUES

The issue presented is:

(1) whether claims 1, 3-4, 6, 8-9, 11, and 13-14 are unpatentable under 35 U.S.C. § 102(b), and

(2) whether claims 2, 7, and 12 are unpatentable under 35 U.S.C. § 103(a), and

(3) whether claims 5, 10, and 15 are unpatentable under 35 U.S.C. § 103(a).

VII. GROUPING OF CLAIMS

Appellants hereby state that claims 1-15 form one group.

VIII. ARGUMENTS

A. Summary of the Applied Rejections

The Final Office Action dated March 10, 2005 rejected claims 1, 3-4, 6, 8-9, 11, and 13-14 under 35 U.S.C. § 102(b) as being anticipated by Kionka, and rejected claims 2, 7, and 12 under 35 U.S.C. § 103(a) as being unpatentable over Kionka in view of Hanna et al. (hereinafter "Hanna") and further in view of Chase, Jr., et al. (hereinafter "Chase"); and rejected claims 5, 10 and 15 under 35 U.S.C. 103(a) as being unpatentable over Kionka as applied to claims 4, 9, 14, in view of Hanna. In making the rejections, the Examiner stated:

> **Double Patenting**
> **3. Claims 5, 10, 15 are provisionally rejected under judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 6, and 18 of copending Application No. 09/903937 (hereinafter '937). Although the conflicting claims are not identical, they are not patentably distinct from each other because of the follow observations. ...**

5. Claims 1, 3-4, 6, 8-9, 11, and 13-14 are rejected under 35 U.S.C. 102(b) as being anticipated by Kionka, USPN: 5,361,357 (hereinafter Kionka).

As per claim 1, Kionka discloses a method for minimizing the cycle time when compiling a program in a computer system (e.g., optimizing, efficient compilation – col. 1, lines 7-25), the program including a plurality of directories (e.g. Fig. 2a) and each of the directories including a code file; the method comprising:

providing a master array of directories of the program (e.g. abstracted tree register 48- col. 6, lines 21-68; Fig. 2C), wherein the master array lists the dependencies of the directories, providing a code change to the program to provide an updated program (e.g. super Makefile; modified smf-col. 7-8; groups of Makefiles out-of-date, doOneMakefile- col. 11-12- Note: compounding individual Make into an updated one Make file is updating with code change;)

providing associated dependency changes to the master array to provide an updated master array (e.g. are to be updated-col. 6, lines 60-68); and

But Kionka does not explicitly disclose compiling the updated program utilizing the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories. However, Kionka discloses a equivalent of a master Makefile integrating all the individual directory Makefiles (e.g. col. 6, lines 31-41; Appendix – col. 7-26); hence has implicitly disclosed compiling of an updated and master Makefile using the order of dependencies as adjusted from integrating the plurality of directories as set forth in the individual Makefiles. Therefore, the limitation is disclosed.

As per claim 3, Kionka discloses that the associated dependencies changes are provided via a directory update mechanism (e.g....are to be updated – col. 6, lines 42-68; col. 7-8; col. 11-12).

As per claim 4, Kionka discloses the steps of:

providing an array of dependency changes (e.g. Directory Description register 50 – Fig. 2a-c; registers 40, 46, 56 – Fig. 1; col. 5, lines 60-67); and

merging the dependency changes array with a master array of changes (e.g. Abstract Tree Register 48, Output file Register 54-Fig. 1).

As per claim 6, Kionka discloses a system for minimizing the cycle time when compiling a program in a computer system, the program including a plurality of directories and each of the directories including a code file, the method comprises the steps of

providing (a master array),

providing (an updated program);

providing (dependencies changes...updated master array); and

compiling (the updated program... an ordered manner); all of which steps having been addressed in claim 1.

As per claims 8-9, these claims correspond to claims 3-4; hence are rejected using the same rejection as set forth therein, respectively.

As per claim 11, this is the computer-readable medium version of method claim 1, including the same step limitations; hence is rejected using the corresponding rejection as set forth therein.

As per claims 13-14, these claims correspond to claims 3-4; hence are rejected using the same rejection as set forth therein, respectively.

7. Claims 2, 7, 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kionka, USPN: 5,361,357 (hereinafter Kionka), as applied to claims 1, 6, 11; in view of Hanna et al., USPN: 5,748,961 (hereinafter Hanna), and further in view of Chase, Jr., et al., USPN: 4,951,192 (hereinafter Chase).

As per claim 2, Kionka does not explicitly teach a scheduler being utilized to compile the update program, wherein the scheduler receives the dependency changes and a list of processors from a processor array. Kionka, however,

mentions about complexity of a system with a larger amount of dependencies and LAN connectivity with more than one processors to distribute storage burden (e.g., Fig. 1). Hanna, in a method to optimize compiling and expedite the object code building in a large system, using a Makefile (ch. 9.2-col. 41) in conjunction with build models or a tree structure of directory models (e.g. Fig. 1a-b) analogous to a master array of directories, teaches a schedule process as to how to distribute the code building process (ch. 10.6-col. 45). The distribution of resources and tasks in a multi-computing network or complex interconnected system using a resource scheduler as suggested by Hanna is further enhanced via the teachings by Chase to provide parallel compilation. Chase, in a system to configure a large software system analogous to Kionka or Hanna, discloses a scheduler to choose and assign available or most suitable processors from a displayed list to compile the buildable components (e.g. col. 1, line 67 to col. 2, line 25). It would have been obvious for one of ordinary skill in the art at the time the invention was made to enhance the complex system connecting a plural computer for handling resources as suggested by Kionka and Hanna so that a scheduling process as taught by Hanna can further be used in compiling task assignment via selection from a plurality of processors being listed as taught by Chase because that way more resources can be distributed and tasks can be imparted separately alleviating thereby the condition of dependency between subsystem task that might otherwise be a limiting factor to expediting the building process and product delivery of a large software system (see Chase: impairs productivity, independent components... parallel – col. 1-2).
As per claims 7 and 12, these are claims corresponding to claim 2, and are rejected using the same rationale as set forth therein.

8. Claims 5, 10 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kionka, USPN: 5,361,357, as applied to claims 4, 9, 14; in view of Hanna et al., USPN: 5,748,961.

As per claim 5, Kionka discloses the steps of:

obtaining a dependency change from the dependency changes array (e.g. step 29-Fig. 2a-c; Input: %rels dependencies for each target – col. 19-20; 21-22 – Note: code change in Makefile is equivalent to obtaining a dependency change, a set of dependent elements being sequences into a list, when processing the directories);

determining whether the dependency change is in a directory in the master array (e.g. exists, die (Not implemented), $fake_top_name not needed-col. 21-22- Note: making adjustment in the master Makefile is equivalent to implementing changes being reflected in the directory element comprising the master array);

(i) updating the directory in the master array of the dependency change in a directory of the master array (e.g. col. 6, lines 60-67);

(iii) determining if there is another dependency change in the dependency changes array after step (i) (see Fig. 2a-c; Appendix code with foreach); and

repeating steps (i) and (iii) until all dependency changes have been obtained from the dependency change array.

But Kionka does not explicitly disclose the step of (ii) adding dependency change to the master array in a new directory if the dependency change is not in a directory of the master array; nor does Kionka disclose repeating of steps (i), (ii) and (iii).

However, Kionka discloses that the master array includes each directory files relationship as those directories are sequentially inputted for processing (e.g. Fig. 2a-c and related text) and teaches a the list of added dependent elements pertinent to the file directory structures being inputted to form the final master structure (e.g. @new_list = shift (); push (@new_list, $top_target – col. 21-22; Fig. 2a, step 31); hence has suggested the idea of creating a list of elements coming

from a specific directory and of adding a list dependency changes to the master array if the change in the array/list of dependency change has not yet been included in the master array (Note: list of dependency element, e.g. $rels Input: %rels dependencies for each target-col. 21-22, is equivalent to array being hierarchized according to priority and rules of a Makefile). Thus, the concept of adding a array of dependency elements or directory-related elements (re Fig. 2a, step 31) into a grouping or any hierarchy of elements being stored and arrayed is suggested, i.e., a hierarchy of dependency being grouped and arrayed by its directory root. Further, Hanna, in the system as mentioned in claim 1, supports grouping of elements into the models in terms of directories (e.g. Fig. 1b).

Based on the teachings by Hanna and on the suggested concepts and master array upgrade teachings by Kionka from above, it would have been obvious for one skill in the art at the time the invention was made to provide the creation of a new change directory within the master array as suggested by Hanna or Kionka so as to include successive dependency changes or array of dependency elements into such form of directory grouping as suggested above and provide the repeating of steps from (i) to (iii) because generating a set of dependent elements being added to a directory, a separate grouping structure or a flattened hierarchy of elements would provide a better view to the abstract tree, i.e. the global master array, since this master array is purposed for enabling a facilitated perception of a more complex system wherein dependency of directories being integrated in the final build is subjected to continual and dynamic updates or adjustments.

As per claims 10 and 15, these are claims corresponding to claim 5, and are rejected using the same rationale as set forth therein.

The Examiner stated the following in response to the previous arguments against these rejections:

Applicant's arguments filed 10/19/05 have been fully considered but they are not persuasive. Following are Examiner's observations in regard thereto.

Rejection 35 U.S.C. § 102:

(A) Applicant has submitted that Kionka's updates to directories as cited in the Office Action are updates to files and files dependencies are not the same as directories dependencies (Appl. Rmrks, 2nd para, pg. 11). The limitation recited as 'dependencies of directories' is not elaborated sufficiently in the claim to enforce that interdependent files of a directory are precluded from being what is construed as 'directories dependencies'. As construed by broad reasonable interpretation, a file is inherent to a directory path and dependency of a directory path files; and updates of such file along with the role of an abstracted structure responsible for reflecting the changes to those files (as cited in the rejection) read on 'providing associated dependency changes ... updated master array'. There are not sufficient details in the claim to distinguish a Master Array from what is recited in Kionka's teaching of a update tree, nor is there enough clarification in the claim that dictates that such array listing of dependencies of directories would be necessary different from the abstracted tree by Kionka because dependency of directories is not construed as different from dependency of directory files inherently listed in a path being one integral part of a directory. Hence, the arguments either read the specifications into the claims or amount to alleged remarks that fail to convince why Kionska's teaching would not fulfill what is explicitly recited.

Rejection 35 U.S.C. § 103:

(B) Applicant has submitted that the rejections (Appl. Rmrks, 2nd para, pg. 14) are disagreed upon based on the arguments set forth for the independent claims 1,

6, and 11. These rejections will stand as set forth on the grounds of the counter-arguments in section A above.

Appellants respectfully request that the Board reverse the Examiner's final rejection of the pending Claims.

B. The Cited Prior Art

Kionka describes a method and apparatus for optimizing the sequencing and time requirements for compiling large sets of source code residing in multiple hierarchical file directories using an abstracted logical description of the hierarchical file relations existing between directories. The system consists of a logic processor working in concert with input and output file registers, a match register, and an abstracted tree register for the purpose of creating a identifying, comparing, and sequencing file names in a final description of the global directory. The method iteratively identifies the primary input files and the intermediate input files for a given output file for each of a series of directories, inverts the casual relationship between the output file and its intermediary input files, and accumulates and stores these relationships in a sequential manner for subsequent use.

Hanna describes a method and apparatus for compiling and linking modules of computer code in a large software system. The software system is defined by a tree of system models, which are written in a functional language. During a build of the software system, the functions are interpreted and the results of the expensive expressions are cached. Each function is examined before interpretation to see if it has been evaluated before. If a function has already been evaluated, the cached result is retrieved by the evaluator and the time, which would have been spent re-evaluating the function is saved.

Chase describes a software configuration management system that uses a network-computing environment to build large software systems in parallel. A configuration manager assigns the compilation of buildable components of a software system to different processors in the network. Buildable components are assigned in order, according to dependencies between components, independent components taking precedence. Processors are chosen according to the amount of idle time during a sampled time segment. A display provides processor compilation status messages for each compilation discrete from status messages of other compilations. A continuously updated overall status report of the system being built is simultaneously displayed with, but segregated from, the compilation status messages.

C. Claims Are Not Unpatentable Under 35 U.S.C. § 102(b) and § 103(a)

The present invention provides a method and system for minimizing the cycle time when compiling a program in a computer system, where a program includes a plurality of directories and each of the directories includes a code file. In accordance with the present invention, the method includes: (a) providing a master array of directories of the program, wherein the master array lists the dependencies of the directories; (b) providing a code change to the program to provide an updated program; (c) providing associated dependency changes to the master array to provide an updated master array; and (d) compiling the updated program utilizing the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories. As a result, compile cycle time for large programs is significantly reduced. A second advantage is that since the dependencies are updated substantially simultaneously with code changes, there are minimal dependency violations and therefore few deadlocks. The cited references do not teach or suggest these features, as discussed below.

With regard to the 102(b) rejections, Applicant respectfully submits that Kionka does not teach the combination of "providing a master array of directories of the program, wherein the master array lists the dependencies of the directories," and "providing associated dependency changes to the master array to provide an updated master array," as recited in independent claims 1, 6, and 11. The Examiner has referred to updates in column 6, lines 60-68, of Kionka. However, the updates of Kionka are not updates to directories as in the present invention but are instead updates to files. Kionka explicitly teaches that the "various files in the different directories are to be updated" (column 6, lines 64-66). In other words, while the files of Kionka are in directories, the files are not the same as directories. Accordingly, the file dependencies of Kionka are different from the directory dependencies of the present invention.

Applicant agrees with the Examiner that Kionka does not explicitly disclose "compiling the updated program utilizing the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories," as recited in independent claims 1, 6, and 11. The Examiner has asserted that this step is implicitly disclosed and has referred to a "master Makefile." However, Kionka does not describe a "master" Makefile, and the term "makefile" is not directed to making directories but is instead directed to making files. Kionka describes a rule that defines a "causal relationship between the files," where the "rule for the files in an abstracted tree will often by "make [filename]" (column 6, lines 31-41). Kionka further states that the abstracted tree is used "as input file describing the sequence by which the various files in the different directories are to be updated (column 6, lines 60-66). Accordingly, any compiling in Kionka is not based upon dependencies of directories as in the present invention but is instead based upon dependencies of files.

Therefore, Kionka does not teach or suggest the combination of steps as recited in independent claims 1, 6, and 11. Accordingly, claims 1, 6, and 11 are allowable over Kionka.

Dependent claims 3-4, 8-9, and 13-14 depend from independent claims 1, 6, and 11, respectively. Accordingly, the above-articulated arguments related to independent claims 1, 6, and 11 apply with equal force to claims 3-4, 8-9, and 13-14, which are thus allowable over the cited references for at least the same reasons as claims 1, 6, and 11.

With regard to the 103(a) rejections, Applicant respectfully disagrees with the Examiner's rejections. Dependent claims 2, 5, 7, 10, 12, and 15 depend from independent claims 1, 6, and 11, respectively. Accordingly, the above-articulated arguments related to independent claims 1, 6, and 11 apply with equal force to claims 2, 5, 7, 10, 12, and 15, which are thus allowable over the cited reference for at least the same reasons as claims 1, 6, and 11.

With regard to the double patenting rejection, Applicant respectfully defers submission of a terminal disclaimer to overcome the rejection upon determinability of allowance of the claims in view of the prior art.

In view of the foregoing, Applicant respectfully submits that the recited invention is not taught, shown, or suggested by the cited art.

Accordingly, Appellants respectfully request withdrawal of the rejections under 35 U.S.C. 102(b) and 35 U.S.C. 102(b), and Appellants respectfully requests that the Board reverse the final rejection of Claims.

E. Summary of Arguments

For all the foregoing reasons, it is respectfully submitted that Claims 1-15 (all the Claims presently in the application) are patentable for defining subject matter, which would not have been unpatentable under 35 U.S.C. §102(b) and §103(a) at the time the subject matter was invented. Thus, Appellants respectfully request that the Board reverse the rejection of all the appealed Claims and find each of these Claims allowable.

Note: For convenience of detachment without disturbing the integrity of the remainder of pages of this Appeal Brief, Appellants' "APPENDIX" section is contained on separate sheets following the signatory portion of this Appeal Brief.

This Brief is being submitted in triplicate, and authorization for payment of the required Brief fee is contained in the cover letter for this Brief. Please charge any fee that may be necessary for the continued pendency of this application to Deposit Account No.

Respectfully submitted,
SAWYER LAW GROUP LLP

July 19, 2005
Date

Joseph A. Sawyer, Jr.
Attorney for Applicant(s)
Reg. No. 30,801
(650) 493-4540

IX. APPENDIX

1. (Original) A method for minimizing the cycle time when compiling a program in a computer system, the program including a plurality of directories and each of the directories including a code file; the method comprises the steps of:

(a) providing a master array of directories of the program, wherein the master array lists the dependencies of the directories,

(b) providing a code change to the program to provide an updated program;

(c) providing associated dependency changes to the master array to provide an updated master array; and

(d) compiling the updated program utilizing the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories.

2. (Original) The method of claim 1 wherein a scheduler is utilized to compile the updated program, wherein the scheduler receives the dependency changes and a list of processors from a processor array.

3. (Original) The method of claim 1 wherein the associated dependencies changes are provided (c) via a directory update mechanism.

4. (Original) The method of claim 3 wherein the providing an update mechanism step (c) further comprises the steps of:

(c1) providing an array of dependency changes; and

(c2) merging the dependency changes array with a master array of changes.

5. (Original) The method of claim 4 wherein the merging step (c2) comprises the steps of:

(c21) obtaining a dependency change from the dependency changes array;

(c22) determining whether the dependency change is in a directory in the master array;

(c23) updating the directory in the master array of the dependency change in a directory of the master array;

(c24) adding dependency change to the master array in a new directory if the dependency change is not in a directory of the master array;

(c25) determining if there is another dependency change in the dependency changes array after either step (c23) or step (c24); and

(c26) repeating steps (c21) – (c25) until all dependency changes have been obtained from the dependency change array.

6. (Original) A system for minimizing the cycle time when compiling a program in a computer system, the program including a plurality of directories and each of the directories including a code file, the method comprises the steps of

(a) providing a master array of directories of the program, wherein the master array lists the dependencies of the directories,

(b) providing a code change to the program to provide an updated program;

(c) providing associated dependencies changes to the master array to provide an updated master array; and

(d) compiling the updated program utilizing of the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories.

7. (Original) The system of claim 6 wherein a scheduler is utilized to compile the updated program, wherein the scheduler receives the dependency changes and a list of processors from a processor array.

8. (Original) The system of claim 6 wherein the associated dependencies changes are provided (c) via a directory update mechanism.

9. (Original) The system of claim 8 wherein the providing an update mechanism step (c) further comprises the steps of:

(c1) providing an array of dependency changes; and

(c2) merging the dependency changes array with a master array of changes.

10. (Original) The system of claim 9 wherein the merging step (c2) comprises the steps of:

(c21) obtaining a dependency change from the dependency changes array;

(c22) determining whether the dependency change is in a directory in the master array;

(c23) updating the directory in the master array of the dependency change in a directory of the master array;

(c24) adding dependency change to the master array in a new directory if the dependency change is not in a directory of the master array;

(c25) determining if there is another dependency change in the dependency changes array after either step (c23) or step (c24); and

(c26) repeating steps (c21) – (c25) until all dependency changes have been obtained from the dependency change array.

11. (Original) A computer readable medium for minimizing the cycle time when compiling a program in a computer system, the program including a plurality of directories and each of the directories including a code file, the method comprises the steps of

(a) providing a master array of directories of the program, wherein the master array lists the dependencies of the directories,

(b) providing a code change to the program to provide an updated program;

(c) providing associated dependencies changes to the master array to provide an updated master array; and

(d) compiling the updated program utilizing the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories.

12. (Original) The computer readable medium of claim 11 wherein a scheduler is utilized to compile the updated program, wherein the scheduler receives the dependency changes and a list of processors from a processor array.

13. (Original) The computer readable medium of claim 11 wherein the associated dependencies changes are provided (c) via a directory update mechanism.

14. (Original) The computer readable medium of claim 13 wherein the providing an update mechanism step (c) further comprises the steps of:

(c1) providing an array of dependency changes; and

(c2) merging the dependency changes array with a master array of changes.

15. (Original) The computer readable medium of claim 14 wherein the merging step (c2) comprises the steps of:

(c21) obtaining a dependency change from the dependency changes array;

(c22) determining whether the dependency change is in a directory in the master array;

(c23) updating the directory in the master array of the dependency change in a directory of the master array;

(c24) adding dependency change to the master array in a new directory if the dependency change is not in a directory of the master array;

(c25) determining if there is another dependency change in the dependency changes array after either step (c23) or step (c24); and

(c26) repeating steps (c21) – (c25) until all dependency changes have been obtained from the dependency change array.